

Plurality Consensus via Shuffling: Lessons Learned from Load Balancing

Petra Berenbrink¹, Tom Friedetzky², Peter Kling¹, Frederik Mallmann-Trenn^{1,3}, and Chris Wastell²

¹Simon Fraser University, Burnaby, Canada

²Durham University, Durham, U.K.

³École normale supérieure, Paris, France

Abstract

We consider *plurality consensus* in a network of n nodes. Initially, each node has one of k opinions. The nodes execute a (randomized) distributed protocol to agree on the *plurality opinion* (the opinion initially supported by the most nodes). Nodes in such networks are often quite cheap and simple, and hence one seeks protocols that are not only fast but also simple and space efficient. Typically, protocols depend heavily on the employed communication mechanism, which ranges from sequential (only one pair of nodes communicates at any time) to fully parallel (all nodes communicate with all their neighbors at once) communication and everything in-between.

We propose a framework to design protocols for a multitude of communication mechanisms. We introduce protocols that solve the plurality consensus problem and are with probability $1 - o(1)$ both time and space efficient. Our protocols are based on an interesting relationship between plurality consensus and distributed load balancing. This relationship allows us to design protocols that generalize the state of the art for a large range of problem parameters. In particular, we obtain the same bounds as the recent result of [3] (who consider only two opinions on a clique) using a much simpler protocol that generalizes naturally to general graphs and multiple opinions.

1 Introduction

The goal of the *plurality consensus* problem is to find the so-called *plurality opinion* (i.e., the opinion that is initially supported by the largest subset of nodes) in a network G where, initially, each of the n nodes has one of k opinions. Applications for this problem include Distributed Computing [18, 32, 33], Social Networks [31, 15, 30], as well as biological interactions [14, 13]. All these areas typically demand both very simple and space-efficient protocols. Communication models, however, can vary from anything between simple sequential communication with a single neighbor (often used in biological settings as a simple variant of asynchronous communication [6]) to fully parallel communication where all nodes communicate with all their neighbors simultaneously (like broadcasting models in distributed computing). This diversity turns out to be a major bottleneck in algorithm design, since protocols (and their analysis) depend to a large part on the employed communication mechanism.

In this paper we present two simple plurality consensus protocols called SHUFFLE and BALANCE. Both protocols work in a very general communication model which uses discrete rounds. The communication partners are determined by a (possibly randomized) sequence $(\mathbf{M}_t)_{t \leq N}$ of *communication matrices*, where we assume¹ N to be some arbitrary large polynomial in n . That is, nodes u and v can communicate in round t if and only if $\mathbf{M}_t[u, v] = 1$. In that case, we call the edge $\{u, v\}$ *active*. Our results allow for a wide class of communication patterns (which can even vary over time) as long as the communication matrices have certain “smoothing” properties (cf. Section 2). These smoothing properties are inspired by similar smoothing properties used by [35] for load balancing in the dimension exchange model. In fact, load balancing is the

¹For simplicity and without loss of generality; our protocols run in polynomial time in all considered models.

source of inspiration for our protocols. Initially, each node creates a suitably chosen number of tokens labeled with its own opinion. Our BALANCE protocol then simply performs discrete load balancing on these tokens, allowing each node to get an estimate on the total number of tokens for each opinion. The SHUFFLE protocol keeps the number of tokens on every node fixed, but shuffles tokens between communication partners. By keeping track of how many tokens of their own opinion (label) were exchanged in total, nodes gain an estimate on the total (global) number of such tokens. Together with a simple broadcast routine, the nodes are able to determine the plurality opinion.

The run time of our protocols is the smallest time t for which all nodes have stabilized on the plurality opinion. That is, all nodes have determined the plurality opinion and will not change. This time depends on the network G , the communication pattern $(M_t)_{t \leq N}$, and the initial bias towards the plurality opinion (cf. Section 2). For both protocols we show a strong correlation between their run time and the mixing time of certain random walks and the (related) *smoothing time*, both of which are used in the analysis of recent load balancing results [35]. To give some more concrete examples of our results, let $T := O(\log n / (1 - \lambda_2))$, where $1 - \lambda_2$ is the spectral gap of G . If the bias is sufficiently high, then both our protocols SHUFFLE and BALANCE determine the plurality opinion in time (a) $n \cdot T$ in the *sequential model* (only one pair of nodes communicates per time step); (b) $d \cdot T$ in the *balancing circuit model* (communication partners are chosen according to d (deterministic) perfect matchings in a round-robin fashion); and (c) T in the *diffusion model* (all nodes communicate with all their neighbors at once). To the best of our knowledge, these match the best known bounds in the corresponding models. For an arbitrary bias, the protocols differ in their time and space requirements. More details about our results can be found in Section 1.2.

1.1 Related Work

The subsequent discussion focuses on distributed models for large networks, where nodes are typically assumed to be very simple, and efficiency is measured in both time and space. Most results depend on the *initial bias* $\alpha := \frac{n_1 - n_2}{n} \in [1/n, 1]$, where n_1 and n_2 denote the number of nodes with the most common and second most common opinions, respectively. The special case of plurality consensus with $k = 2$ is often referred to as *majority voting* or *binary consensus*. Similar to [8], we use the term *plurality* (instead of majority) to highlight that, for $k > 2$, the opinion supported by the largest subset of nodes might be far from an (absolute) majority.

Population Protocols. The first major line of work on majority voting considers *population protocols*. Here, nodes are modelled as finite state machines with a small state space. Communication partners are chosen either adversarial or randomly. See [6, 5] for a more detailed model description. [4] propose a 3-state (i.e., constant memory) population protocol for majority voting (i.e., $k = 2$) on the clique to model the mixing behavior of molecules. We refer to their communication model as the *sequential model*: each time step, an edge is chosen uniformly at random, such that only one pair of nodes communicates. If the initial bias α is $\omega(\log n / \sqrt{n})$, their protocol lets all nodes agree (w.h.p.) on the majority opinion in $O(n \cdot \log n)$ steps. [29] show that this 3-state protocol fails on general graphs in that there are infinitely many graphs on which it returns the minority opinion or has exponential run time. They also provide a 4-state protocol for *exact* majority voting, which *always* returns the majority opinion (independent of α) in time $O(n^6)$ on arbitrary graphs and in time $O\left(\frac{\log n}{\alpha} \cdot n^2\right)$ on the clique. This result is optimal in that no population protocol for exact majority can have fewer than four states. A very recent result is due to [3]. They give a sophisticated (if slightly complicated) protocol for $k = 2$ on the clique in the sequential model. It solves exact majority and has (w.h.p.) *parallel run time*² $O\left(\frac{\log^2 n}{s \cdot \alpha} + \log^2 n \cdot \log s\right)$. Here, s is the number of states and must be in the range $s = O(n)$ and $s = \Omega(\log n \cdot \log \log n)$.

Pull Voting. The second major research line on plurality consensus has its roots in gossiping and rumor spreading. Communication in these models is often restricted to pull requests, where nodes can query other nodes' opinions and use a simple rule to update their own opinion (note that the 3-state protocol from [4] fits

²Parallel run time in the sequential model is the number of (sequential) time steps divided by n . This is a typical measure for population protocols and based on the intuition that, in expectation, each node communicates with one neighbor within n time steps.

into this model). See [32] for a slightly dated but thorough survey. In a recent result, [17] consider a voting process for two opinions on arbitrary d -regular graphs. They pull the opinion of two random neighbors and, if the pulled opinions are identical, adopt it. For random d -regular graphs, (w.h.p.) all nodes agree after $O(\log n)$ steps on the plurality opinion (provided that $\alpha = \Omega(\sqrt{1/d + d/n})$). For an arbitrary d -regular graph G , they need $\alpha = \Omega(\lambda)$ (where $1 - \lambda_2$ is the spectral gap of G). [9] consider a similar update rule on the clique for k opinions. Here, each node pulls the opinion of three random neighbors and adopts the majority opinion among those three (breaking ties uniformly at random). They need $O(\log k)$ memory bits and prove a tight run time of $\Theta(k \cdot \log n)$ for this protocol (given a sufficiently high bias α). In another recent paper, [8] build upon the idea of the 3-state population protocol from [4]. Using a slightly different time and communication model, they generalize the protocol to k opinions (on the clique). In their model, nodes act in parallel and pull the opinion of a random neighbor each round. Given a memory of $\log k + O(1)$ bits and assuming $k = O((n/\log n)^{1/6})$, they agree (w.h.p.) on the plurality opinion in time $O(k \cdot \log n)$ (given a sufficiently high bias³). Note that, in contrast to all these results, we require our protocols to work for *any* bias, even if it is only by one node (similar to [3]).

Further Models. Aside from the two research lines mentioned above, there is a multitude of related but quite different models. They differ, for example, in the consensus requirement, the time model, or the graph models. This paragraph gives merely a small overview over such model variants. For details, the reader is referred to the corresponding literature.

In one very common variant of the voter model [24, 19, 23, 16, 27, 2, 26, 28], one is interested in the time it takes for the nodes to agree on *some* (arbitrary) opinion. Notable representatives of this flavor are [18, 10]. Both papers consider a consensus variant where the consensus can be on an arbitrary opinion (instead of on the plurality). They have the additional requirement that the agreement is robust even in the presence of adversarial corruptions. Another variant [33] of distributed voting considers the 3-state protocol from [4] (for two opinions on the complete graph), but in a continuous time model. A third variant [1] considers majority voting on special graphs given by a degree sequence. Other protocols such as the one presented in [20] guarantee convergence to the majority opinion. The authors of [20] analyse their protocol for 2 opinions.

Load Balancing. While our problem is quite different from load balancing, our results use techniques from and show interesting connections to certain types of discrete load balancers⁴. In discrete load balancing, each node starts with an arbitrary number of tokens. Each time step, nodes can exchange load over active edges. The goal is typically to minimize the *discrepancy* (the maximum load difference between any pair of nodes), and K denotes the initial discrepancy. The following results for d -regular graphs hold due to [35, 34]: The discrepancy can be reduced to (a) a constant in $O(n \cdot \log(Kn)/(1 - \lambda_2))$ time steps in the sequential model, (b) a constant in $O(d \cdot \log(Kn)/(1 - \lambda_2))$ time steps in the balancing circuits model for d perfect matchings, and (c) $O(\sqrt{d} \cdot \log n/(1 - \lambda_2))$ in $O(\log(Kn)/(1 - \lambda_2))$ time steps in the diffusion model.

1.2 Our Contribution

We introduce two protocols for plurality consensus, called SHUFFLE and BALANCE. Both solve plurality consensus in a discrete time model under a diverse set of (randomized or adversarial) communication patterns for an arbitrary non-zero bias. In particular, our very simple BALANCE protocol generalizes the work of [3] threefold: (a) to arbitrary graphs, (b) to an arbitrary number k of opinions, and (c) to more general (and truly parallel) communication models. This generalization to parallel communication models requires a much more careful analysis, since we must deal with additional dependencies. We continue with a detailed description of our results. For both protocols, we give both run time and memory (measured in bits) bounds.

SHUFFLE. Our main result is the SHUFFLE protocol. In the first time step each node generates γ tokens labeled with its initial opinion. During round t , any pair of nodes connected by an active edge (as specified by the communication pattern $(\mathbf{M}_t)_{t \leq N}$) exchanges tokens. We show that SHUFFLE solves plu-

³Their bias definition differs slightly from previous work, requiring $n_1 \geq (1 + \varepsilon)n_2$ for a constant $\varepsilon > 0$.

⁴Similar connections can be drawn to work on averaging (e.g., [25, 12]). However, due to the integrality/memory constraints, our problem is more closely related to (discrete) load balancing as considered in [34, 35].

rality consensus and allows for a trade-off between run time and memory. More exactly, let the number of tokens $\gamma = O(\log n/(\alpha^2 \cdot T))$, where T is a parameter to control the trade-off between memory and run time. Moreover, let t_{mix} be such that any time interval $[t, t + t_{\text{mix}}]$ is ε -smoothing⁵ (cf. Section 2). Then, SHUFFLE ensures that all nodes agree on the plurality opinion in $O(T \cdot t_{\text{mix}})$ rounds (w.h.p.), using $O(\log n/(\alpha^2 T) \cdot \log k + \log(T \cdot t_{\text{mix}}))$ memory bits per node. This implies, for example, that plurality consensus on expanders in the sequential model is achieved in $O(T \cdot n \log n)$ time steps and that nodes require only $O(\log n \cdot \log k/T + \log(Tn))$ memory bits (assuming a constant initial bias). For arbitrary graphs and many natural communication patterns (e.g., communicating with all neighbors in every round or communicating via random matchings), the time for plurality consensus is closely related to the spectral gap of the underlying communication network (cf. Corollary 3). To the best of our knowledge, this is the first plurality protocol (for more than two opinions) that can handle an arbitrary initial bias.

While our protocol is relatively simple, the analysis is much more involved. The idea is to observe a single token only, and to show that, after t_{mix} time steps, the token is (roughly) on any node with the same probability. The main ingredients are Lemmas 9 and 10, a generalization of a result by [35]. These lemmas show that the joint distribution of token locations is negatively correlated, allowing us to derive a suitable Chernoff bound. This is then used to show that, after t_{mix} time steps, each node has a pretty good idea of the total numbers of tokens that are labeled with its opinion. Using a broadcast-like protocol (nodes always forward their guess of the plurality), all nodes can determine the plurality opinion. We believe that the non-trivial generalization of the negative correlation result is interesting in its own right.

BALANCE. The previous protocol (SHUFFLE) allows for a nice trade-off between run time and memory. If the number of opinions is comparatively small, our much simpler BALANCE protocol gives better results. In BALANCE, each node u maintains a k -dimensional load vector. If j denotes u 's initial opinion, the j -th dimension of this load vector is initialized with $\gamma \in \mathbb{N}$ (a sufficiently large value) and any other dimension is initialized with zero. In each time step, all nodes perform a simple, discrete load balancing on each dimension of these load vectors. Our results imply, for example, that plurality consensus on expanders in the sequential model is achieved in only $O(n \cdot \log n)$ time steps with $O(k)$ memory bits per node (assuming a constant initial bias). In the setting considered by [3] (but for arbitrary k instead of $k = 2$), BALANCE achieves plurality consensus in time $O(n \cdot \log n)$ and uses $O(\log(1/\alpha) \cdot k)$ bits per node (Corollary 13). This not only improves⁶ by a logarithmic factor over [3] (who consider $k = 2$), but generalizes the results to $k > 2$ via a much simpler protocol (although both protocols are similar in spirit).

2 Model & General Definitions

We consider an undirected graph $G = (V, E)$ of $n \in \mathbb{N}$ nodes and let $1 - \lambda_2$ denote the eigenvalue (or spectral) gap of G . Note that $1 - \lambda_2$ is constant for expanders and the clique. Each node u is assigned an *opinion* $o_u \in \{1, 2, \dots, k\}$. For $i \in \{1, 2, \dots, k\}$, we use $n_i \in \mathbb{N}$ to denote the number of nodes which have initially opinion i . Without loss of generality (w.l.o.g), we assume $n_1 > n_2 \geq \dots \geq n_k$, such that 1 is the opinion that is initially supported by the largest subset of nodes. We also say that 1 is the *plurality opinion*. The value $\alpha := \frac{n_1 - n_2}{n} \in [1/n, 1]$ denotes the *initial bias* towards the plurality opinion. In the *plurality consensus problem*, the goal is to design simple, distributed protocols that let all nodes agree on the plurality opinion. Time is measured in discrete rounds, such that the (randomized) run time of our protocols is the number of rounds it takes until all nodes are aware of the plurality opinion. As a second quality measure, we consider the total number of memory bits per node that are required by our protocols⁷. All our statements and proofs assume n to be sufficiently large.

Communication Model. In any given round, two nodes u and v can communicate if and only if the edge between u and v is *active*. We use M_t to denote the symmetric *communication matrix* at time t , where $M_t[u, v] = M_t[v, u] = 1$ if $\{u, v\}$ is active and $M_t[u, v] = M_t[v, u] = 0$ otherwise. We assume

⁵Intuitively, this means that the communication pattern has good load balancing properties during any time window of length t_{mix} . This coincides with is the worst-case mixing time of a lazy random walk on active edges.

⁶Note that the bounds in [3] are stated in parallel time, which is simply the normal run time divided by n .

⁷Literature more closely related to biological settings often uses the number of states $s \in \mathbb{N}$ a node can have to measure the space requirement of protocols (e.g., [3]). Such protocols require $\lceil \log s \rceil$ bits per node.

(w.l.o.g) $\mathbf{M}_t[u, u] = 1$ (allowing nodes to “communicate” with themselves). Typically, the sequence $\mathbf{M} = (\mathbf{M}_t)_{t \in \mathbb{N}}$ of communication matrices (the *communication pattern*) is either randomized or adversarial, and our statements merely require that \mathbf{M} satisfies certain smoothing properties (see below). For the ease of presentation, we restrict ourselves to polynomial number of time steps and consider only communication patterns $\mathbf{M} = (\mathbf{M}_t)_{t \leq N}$ where $N = N(n)$ is an arbitrarily large polynomial. Let us briefly mention some natural and common communication models covered by such patterns:

- *Diffusion Model*: All edges of the graph are permanently activated.
- *Random matching model*: In every round t , the active edges are given by a random matching. We require that random matchings from different rounds are mutually independent⁸. While we do not restrict the exact way how the matching is chosen, results for the random matching model dependent on the parameter $p_{\min} := \min_{t \in \mathbb{N}, \{u, v\} \in E} \Pr(\mathbf{M}_t[u, v] = 1)$.
- *Balancing Circuit Model*: There are d perfect matchings $\mathbf{M}_0, \mathbf{M}_1, \dots, \mathbf{M}_{d-1}$ given. They are used in a round-robin fashion, such that for $t \geq d$ we have $\mathbf{M}_t = \mathbf{M}_{t \bmod d}$.
- *Sequential Model*: In every round t , one edge $\{u, v\} \in E$ is chosen uniformly at random and activated (i.e., \mathbf{M}_t has exactly 4 non-zero entries).

Notation. We use $\|\mathbf{x}\|_\ell$ to denote the ℓ -norm of vector \mathbf{x} , where the ∞ -norm is the vector’s maximum absolute entry. In general, bold font indicates vectors and matrices, and we use $x(i)$ to refer to the i -th component of vector \mathbf{x} . The *discrepancy* of a vector \mathbf{x} is defined as $\text{disc}(\mathbf{x}) := \max_i x(i) - \min_i x(i)$. For a natural number $i \in \mathbb{N}$, we define $[i] := \{1, 2, \dots, i\}$ as the set of the first i integers. We use $\log x$ to denote the binary logarithm of $x \in \mathbb{R}_{>0}$. We write $a \mid b$ if a divides b . For any node $u \in V$, we use $d(u)$ to denote u ’s degree in G and $d_t(u) := \sum_v \mathbf{M}_t[u, v]$ to denote its *active degree* at time t (i.e., its degree when restricted to active edges). Similarly, $N(u)$ and $N_t(u)$ are used to refer u ’s (active) neighborhood. Moreover, let $\Delta := \max_{t, u} d_t(u)$ be the maximum active degree of any node at any time in the given communication pattern. We assume knowledge of Δ , which merely means we assume that the nodes are aware of the communication model.

Smoothing Property. The run time of our protocols is closely related to the run time (“smoothing time”) of diffusion load balancing algorithms, which in turn is a function of the mixing time of a random walk on G . More exactly, we consider a random walk on G that is restricted to the active edges in each time step. As indicated in Section 1.2, this random walk should converge towards the uniform distribution over the nodes of G . This leads to the following definition of the random walk’s transition matrices \mathbf{P}_t based on the communication matrices \mathbf{M}_t :

$$\mathbf{P}_t[u, v] := \begin{cases} \frac{1}{2\Delta} & \text{if } \mathbf{M}_t[u, v] = 1 \text{ and } u \neq v, \\ 1 - \frac{d_t(u)}{2\Delta} & \text{if } \mathbf{M}_t[u, v] = 1 \text{ and } u = v, \\ 0 & \text{if } \mathbf{M}_t[u, v] = 0. \end{cases} \quad (1)$$

Obviously, \mathbf{P}_t is doubly stochastic for all $t \in \mathbb{N}$. Moreover, note that the random walk is trivial in any matching-based model, while we get $\mathbf{P}_t[u, v] = \frac{1}{2d}$ for every edge $\{u, v\} \in E$ in the diffusion model on a d -regular graph. We are now ready to define the required smoothing property.

Definition 1 (ε -smoothing). *Consider a fixed sequence $(\mathbf{M}_t)_{t \leq N}$ of communication matrices and a time interval $[t_1, t_2]$. We say $[t_1, t_2]$ is ε -smoothing (under $(\mathbf{M}_t)_{t \leq N}$) if for any non-negative vector \mathbf{x} with $\|\mathbf{x}\|_\infty = 1$ it holds that $\text{disc}(\mathbf{x} \cdot \prod_{t=t_1}^{t_2} \mathbf{P}_t) \leq \varepsilon$. Moreover, we define the mixing time $t_{\text{mix}}(\varepsilon)$ as the smallest number of steps such that any time window of length $t_{\text{mix}}(\varepsilon)$ is ε -smoothing. That is,*

$$t_{\text{mix}}(\varepsilon) := \min \{ t' \mid \forall t \in \mathbb{N}: [t, t + t'] \text{ is } \varepsilon\text{-smoothing} \}. \quad (2)$$

Note that the mixing time can be seen as the worst-case time required by a random walk to get “close” to the uniform distribution. If the parameter ε is not explicitly stated, we consider $t_{\text{mix}} := t_{\text{mix}}(n^{-5})$. To simplify the description of our protocols we assume that all nodes know t_{mix} . This is without loss of generality, as we can “guess” the mixing time by a standard doubling-approach. Note that t_{mix} depends on the sequence $(\mathbf{M}_t)_{t \leq N}$ of communication matrices.

⁸Note that there are several simple, distributed protocols to obtain such matchings [22, 12].

3 Protocol SHUFFLE

Our main result is the following theorem, stating the correctness as well as the time- and space-efficiency of SHUFFLE. A formal description of SHUFFLE can be found in Section 3.1, followed by its analysis in Section 3.2.

Theorem 2. *Let $\alpha = \frac{n_1 - n_2}{n} \in [1/n, 1]$ denotes the initial bias. Consider a fixed communication pattern $(\mathbf{M}_t)_{t \leq N}$ and an arbitrary parameter $T \in \mathbb{N}$. Protocol SHUFFLE ensures that all nodes know the plurality opinion after $O(T \cdot t_{\text{mix}})$ rounds (w.h.p.)⁹ and requires $\left(12 \cdot \frac{\log(n)}{\alpha^2 \cdot T} + 4\right) \cdot \log(k) + 4 \log\left(\frac{12 \cdot \log(n)}{\alpha^2}\right) + \log(T \cdot t_{\text{mix}})$ memory bits per node.*

The parameter T in the theorem statement serves as a lever to trade run time for memory. Since t_{mix} depends on the graph and communication pattern, Theorem 2 might look a bit unwieldy. The following corollary gives a few concrete examples for common communication patterns on general graphs.

Corollary 3. *Let G be an arbitrary d -regular graph. SHUFFLE ensures that all nodes agree on the plurality opinion (w.h.p.) using $\left(12 \cdot \frac{\log(n)}{\alpha^2 \cdot T} + 4\right) \cdot \log(k) + 4 \log\left(\frac{12 \cdot \log(n)}{\alpha^2}\right) + \log(T \cdot t_{\text{mix}})$ bits of memory in time*

- (a) $O\left(T \cdot \frac{\log(n)}{1 - \lambda_2}\right)$ in the diffusion model,
- (b) $O\left(\frac{T}{d \cdot p_{\min}} \cdot \frac{\log(n)}{1 - \lambda_2}\right)$ in the random matching model,
- (c) $O\left(T \cdot d \cdot \frac{\log(n)}{1 - \lambda_2}\right)$ in the balancing circuit model, and
- (d) $O\left(T \cdot n \cdot \frac{\log(n)}{1 - \lambda_2}\right)$ in the sequential model.

3.1 Protocol Description

We continue to explain the SHUFFLE protocol given in Listing 1. Our protocol consists of three parts that are executed in each time step: (a) the *shuffle* part, (b) the *broadcast* part, and (c) the *update* part. Every node u is initialized with $\gamma \in \mathbb{N}$ tokens labeled with u 's opinion o_u . Our protocol sends 2Δ tokens chosen uniformly at random (without replacement) over each edge $\{u, v\} \in E$. Here, $\gamma \geq 2\Delta^2$ is a parameter depending on T and α (to be fixed during the analysis). SHUFFLE maintains the invariant that, at any time, all nodes have exactly γ tokens. In addition to storing the tokens, each node maintains a set of auxiliary variables. The variable c_u is increased during the update part and counts tokens labeled o_u . The variable pair (dom_u, e_u) is a temporary guess of the plurality opinion and its frequency. During the broadcast part, nodes broadcast these pairs, replacing their own pair whenever they observe a pair with higher frequency. Finally, the variable plu_u represents the opinion currently believed to be the plurality opinion. The shuffle and broadcast parts are executed in each time step, while the update part is executed only every t_{mix} time steps¹⁰.

Waiting t_{mix} time steps for each update gives the broadcast enough time to inform all nodes and ensures that the tokens of each opinion are well distributed. The latter implies that, if we consider a node u with opinion $o_u = i$ at time $T \cdot t_{\text{mix}}$, the value c_u is a good estimate of $T \cdot \gamma n_i / n$ (which is maximized for the plurality opinion). When we reset the broadcast (Line 13), the subsequent t_{mix} broadcast steps ensure that all nodes get to know the pair (o_u, c_u) for which c_u is maximal. Thus, if we can ensure that c_u is a good enough approximation of $T \cdot \gamma n_i / n$, all nodes get to know the plurality.

3.2 Analysis of SHUFFLE

Fix a communication pattern $(\mathbf{M}_t)_{t \leq N}$ and an arbitrary parameter $T \in \mathbb{N}$. Remember that $t_{\text{mix}} = t_{\text{mix}}(n^{-5})$ denotes the smallest number such that any time window of length t_{mix} is n^{-5} -smoothing under $(\mathbf{M}_t)_{t \leq N}$. We set the number of tokens stored in each node to $\gamma := \lceil c \cdot \frac{\log n}{\alpha^2 T} \rceil$, where c is a suitable constant. The analysis of SHUFFLE is largely based on Lemma 5, which states that, after $O(T \cdot t_{\text{mix}})$ time steps, the counter values c_u

⁹We say an event happens with high probability (w.h.p.) if its probability is at least $1 - 1/n^c$ for $c \in \mathbb{N}$.

¹⁰It is not essential for the protocol to know the mixing time. Using standard techniques, the protocol can guess the mixing time and grow the guess exponentially.

```

1  for  $\{u, v\} \in E$  with  $M_t[u, v] = 1$ :           {shuffle part}
2      send  $2\Delta$  tokens chosen u.a.r. (without replacement) to  $v$ 
3
4  for  $\{u, v\} \in E$  with  $M_t[u, v] = 1$ :           {broadcast part}
5      send  $(dom_u, e_u)$ 
6      receive  $(dom_v, e_v)$ 
7       $v := w$  with  $e_w \geq e_{w'} \quad \forall w, w' \in N_t(u) \cup \{u\}$ 
8       $(dom_u, e_u) := (dom_v, e_v)$ 
9
10 if  $t \equiv 0 \pmod{t_{\text{mix}}}$ :                       {update part}
11     increase  $c_u$  by the number of tokens labeled  $o_u$  held by  $u$ 
12      $plu_u := dom_u$            {plurality guess: last broadcast's dom. op.}
13      $(dom_u, e_u) := (o_u, c_u)$    {reset broadcast}

```

Listing 1: Protocol SHUFFLE as executed by node u at time t . At time zero, each node u creates γ tokens labeled o_u and sets $c_u := 0$ and $(dom_u, e_u) := (o_u, c_u)$.

can be used to reliably separate the plurality opinion from any other opinion. The main technical difficulty is the huge dependency between the tokens' movements, rendering standard Chernoff-bounds inapplicable.

Instead, we show that certain random variables satisfy the negative regression condition (Lemma 10), which allows us to majorize the token distribution by a random walk (Lemma 9) and to derive the following Chernoff bound.

Lemma 4. *Consider any subset B of tokens, a node $u \in V$, and an integer T . Let $X := \sum_{t \leq T} \sum_{j \in B} X_{j,t}$, where $X_{j,t}$ is 1 if token j is on node u at time $t \cdot t_{\text{mix}}$. With $\mu := (1/n + 1/n^5) \cdot |B| \cdot T$, we have*

$$\Pr(X \geq (1 + \delta) \cdot \mu) \leq e^{\delta^2 \mu / 3}. \quad (3)$$

The lemma's proof is a relatively straightforward consequence of Lemma 9 (which is stated further below) and can be found at the end of this section. Together, these lemmas generalize a result given in [35] to settings where nodes exchange load with more than one neighbor at a time, such that we have to deal with more complex dependencies.

Separating the Plurality via Chernoff

Equipped with the Chernoff bound from Lemma 4, we prove concentration of the counter values and, subsequently, Theorem 2.

Lemma 5. *Let $c \geq 12$. For every time $t \geq c \cdot T \cdot t_{\text{mix}}$ there exist values $\ell_{\top} > \ell_{\perp}$ such that*

- (a) *For all nodes w with $o_w \geq 2$ we have (w.h.p.) $c_w \leq \ell_{\perp}$.*
- (b) *For all nodes v with $o_v = 1$ we have (w.h.p.) $c_v \geq \ell_{\top}$.*

Proof. For two nodes v and w with $o_v = 1$ and $o_w \geq 2$, $\mu_i := (1/n + 1/n^5)c \cdot T \cdot \gamma \cdot n_k$ for all $i \in [k]$, and $\mu' := (1/n + 1/n^5)c \cdot T \cdot \gamma \cdot (n - n_1)$. For $i \in [k]$ define

$$\ell_{\perp}(i) := \mu_i + \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma \frac{n_i}{n}} \quad \text{and}$$

$$\ell_{\top} := T\gamma - \mu' - \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma \frac{n - n_1}{n}}.$$

We set $\ell_{\perp} := \ell_{\perp}(2)$. It is easy to show that $\ell_{\perp} < \ell_{\top}$. Now, let all γn tokens be labeled from 1 to γn . It remains to prove the lemma's statements:

- (a) For the first statement, consider a node w with $o_w \geq 2$ and set $\lambda(o_w) := \ell_{\perp}(o_w) - \mu_{o_w} = \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma \cdot n_{o_w} / n}$. Set the random indicator variable $X_{i,t}$ to be 1 if and only if i is on node w at time t and if i 's label is

o_w . Let $c_w = \sum_{i \in [\gamma n]} \sum_{j \leq T} X_{i,j \cdot t_{\text{mix}}}$. We compute

$$\begin{aligned} \Pr(c_w \geq \ell_\perp) &\leq \Pr(c_w \geq \mu_{o_w} + \lambda(o_w)) \\ &= \Pr\left(c_w \geq \left(1 + \frac{\lambda(o_w)}{\mu_{o_w}}\right) \cdot \mu_{o_w}\right) \\ &\leq \exp\left(-\frac{\lambda^2(o_w)}{3\mu_{o_w}}\right) \leq \exp\left(-\frac{c}{6} \log n\right), \end{aligned} \tag{4}$$

where the last line follows by Lemma 4 applied to $c_w = \sum_{i \in [\gamma n]} \sum_{j \leq T} X_{i,j \cdot t_{\text{mix}}}$ and setting B to the set of all tokens with label o_w . Hence, the claim follows for c large enough after taking the union bound over all $n - n_1 \leq n$ nodes w with $o_w \geq 2$.

- (b) For the lemma's second statement, consider a node v with $o_v = 1$ and set $\lambda' := \mu' - \ell_\top$. Define the random indicator variable $Y_{i,t}$ to be 1 if and only if token i is on node v at time t and if i 's label is not 1. Set $Y = \sum_{j \leq T} \sum_{i \in [\gamma n]} Y_{i,j \cdot t_{\text{mix}}}$ and note that $c_v = T\gamma - Y$. We compute

$$\begin{aligned} \Pr(c_v \leq \ell_\top) &= \Pr(T\gamma - Y \leq \ell_\top) \\ &= \Pr(T\gamma - Y \leq T\gamma - \mu' - \lambda') = \Pr(Y \geq \mu' + \lambda') \\ &= \Pr\left(Y \geq \left(1 + \frac{\lambda'}{\mu'}\right) \cdot \mu'\right) \leq \exp\left(-\frac{\lambda'^2}{3\mu'}\right) \\ &\leq \exp\left(-\frac{c}{6} \log n\right), \end{aligned}$$

where the first inequality follows by Lemma 4 applied to Y and using B to denote the set of all tokens with a label other than 1. Hence, the claim follows for c large enough after taking the union bound over all $n_1 \leq n$ nodes v with $o_u \geq 2$. \square

With Lemma 5, we can now prove our main result:

Proof of Theorem 2. Fix an arbitrary time $t \in [c \cdot T \cdot t_{\text{mix}}, N]$ with $t_{\text{mix}} \mid t$, where c is the constant from the statement of Lemma 5. From Lemma 5 we have that (w.h.p.) the node u with the highest counter c_u has $o_u = 1$ (ties are broken arbitrarily). In the following we condition on $o_u = 1$. We claim that at time $t' = t + t_{\text{mix}}$ all nodes $v \in V$ have $plu_v = 1$. This is because the counters during the ‘‘broadcast part’’ (Lines 4 to 8) propagate the highest counter received after time t . The time τ until all nodes $v \in V$ have $plu_v = 1$ is bounded by the mixing by definition: In order for $[t, t']$ to be $1/n^5$ -smoothing, the random walk starting at u at time t is with probability at least $1/n - 1/n^5$ on node v and, thus, there exists a path from u to v (with respect to the communication matrices). If there is such a path for every node v , the counter of u was also propagated to that v and we have $\tau \leq t_{\text{mix}}$. Consequently, at time t' all nodes have the correct majority opinion. This implies the desired time bound. For the memory requirements, note that each node u stores γ tokens with a label from the set $[k]$ ($\gamma \cdot O(\log k)$ bits), three opinions (its own, its plurality guess, and the dominating opinion; $O(\log k)$ bits), the two counters c_u and e_u and the time step counter. The memory to store the counter c_u and e_u is $O(\gamma T)$. Finally, the time step counter is bounded by $O(\log(T \cdot t_{\text{mix}}))$ bits. Note that it is easy to implement a rolling counter when this counter ‘‘overflows’’. This yields the claimed space bound. \square

Majorizing SHUFFLE by Random Walks

We now turn to the proof of Lemma 4. While our SHUFFLE protocol assumes that 2Δ divides γ , we assume here the slightly weaker requirement that $\mathbf{P}_t[u, v] \cdot \gamma \in \mathbb{N}$ for any $u, v \in V$ and $t \in \mathbb{N}$. To ease the discussion, we consider u as a neighbor of itself and speak of $d_t(u) + 1$ neighbors. For $i \in [d_t(u) + 1]$, let $N_t(u, i) \in V$ denote the i -th neighbor of u (in an arbitrary order). We also need some notation for the shuffle part of our protocol. To this end, consider a node u at time t and let u 's tokens be numbered from 1 to γ . Our assumption on γ allows us to partition the tokens into $d_t(u) + 1$ disjoint subsets (*slots*) $S_i \subseteq [\gamma]$ of size $\mathbf{P}_t[u, v] \cdot \gamma$ each, where $v = N_t(u, i)$. Let $\pi_{t,u} : [\gamma] \rightarrow [\gamma]$ be a random permutation. Token j with $\pi_{t,u}(j) \in S_i$ is sent to u 's i -th neighbor.

Let \mathcal{S} denote our random SHUFFLE process and \mathcal{W} the random walk process in which each of the γn tokens performs an independent random walk according to the random walk matrices $(\mathbf{P}_t)_{t \in \mathbb{N}}$. We use $w_j^{\mathcal{P}}(t)$ to denote the position of token j after t steps of a process \mathcal{P} . Without loss of generality, we assume $w_j^{\mathcal{S}}(0) = w_j^{\mathcal{W}}(0)$ for all tokens j . While there are strong correlations between the tokens' movements in \mathcal{S} (e.g., not all tokens can move to the same neighbor), Lemma 9 shows that these correlations are negative. Before proving Lemma 9 we present the following definitions and auxiliary results that are used in its proof.

Definition 6 (Neg. Regression [21, Def. 21]). *A vector (X_1, X_2, \dots, X_n) of random variables is said to satisfy the negative regression condition if $\mathbb{E}[f(X_l, l \in \mathcal{L}) \mid X_r = x_r, r \in \mathcal{R}]$ is non-increasing in each x_r for any disjoint $\mathcal{L}, \mathcal{R} \subseteq [n]$ and for any non-decreasing function f .*

Lemma 7 ([21, Lemma 26]). *Let (X_1, X_2, \dots, X_n) satisfy the negative regression condition and consider an arbitrary index set $I \subseteq [n]$ as well as any family of non-decreasing functions f_i ($i \in \{I\}$). Then, we have*

$$\mathbb{E} \left[\prod_{i \in I} f_i(X_i) \right] \leq \prod_{i \in I} \mathbb{E}[f_i(X_i)] \quad (5)$$

for any $I \subseteq [n]$ and for any non-decreasing functions f_i .

Claim 8. *Fix a time $t' \in \{0, 1, \dots, t-1\}$ and consider an arbitrary configuration c . Let $\mathcal{E}_{t'}$, X_j and h_j be defined as in the proof of Lemma 9. Then the following identities hold:*

- (a) $\Pr(\mathcal{E}_{t'+1} \mid c(t') = c) = \mathbb{E} \left[\prod_{j \in B} h_j(X_j) \mid c(t') = c \right]$, and
- (b) $\Pr(\mathcal{E}_{t'} \mid c(t') = c) = \prod_{j \in B} \mathbb{E}[h_j(X_j) \mid c(t') = c]$.

Proof. We use the shorthand $d(u_j) = d_{t'+1}(u_j)$. Remember that each X_j indicates to which of the $d(u_j) + 1$ neighbors of u_j (where u_j is considered a neighbor of itself) a token j moves during time step $t' + 1$. Thus, given the configuration $c(t') = c$ immediately before time step $t' + 1$, there is a bijection between any possible configuration $c(t' + 1)$ and outcomes of the random variable vector $\mathbf{X} = (X_j)_{j \in [\gamma n]}$. Let $c_{\mathbf{x}}$ denote the configuration corresponding to a concrete outcome $\mathbf{X} = \mathbf{x} \in [d(u_j) + 1]^{\gamma n}$. Thus, we have $\Pr(c(t' + 1) = c_{\mathbf{x}} \mid c(t') = c) = \Pr(\mathbf{X} = \mathbf{x} \mid c(t') = c)$, and conditioning on $c(t' + 1)$ is equivalent to conditioning on \mathbf{X} and $c(t')$. For the claim's first statement, we calculate

$$\begin{aligned} & \Pr(\mathcal{E}_{t'+1} \mid c(t') = c) \\ &= \sum_{c_{\mathbf{x}}} \Pr(\mathcal{E}_{t'+1} \mid c(t' + 1) = c_{\mathbf{x}}) \cdot \Pr(c(t' + 1) = c_{\mathbf{x}} \mid c(t') = c) \\ &= \sum_{c_{\mathbf{x}}} \prod_{j \in B} \Pr(w_j^{\mathcal{SW}(t'+1)}(t) \in D \mid \mathbf{X} = \mathbf{x}, c(t') = c) \cdot \Pr(\mathbf{X} = \mathbf{x} \mid c(t') = c) \\ &= \sum_{c_{\mathbf{x}}} \prod_{j \in B} h_j(x_j) \cdot \Pr(\mathbf{X} = \mathbf{x} \mid c(t') = c) \\ &= \sum_{\mathbf{x}} \prod_{j \in B} h_j(x_j) \cdot \Pr(\mathbf{X} = \mathbf{x} \mid c(t') = c) = \mathbb{E} \left[\prod_{j \in B} h_j(X_j) \mid c(t') = c \right]. \end{aligned}$$

Here, we first apply the law of total probability. Then, we use the bijection between $c(t' + 1)$ and \mathbf{X} (if $c(t')$ is given) and that the process $\mathcal{SW}(t' + 1)$ consists of independent random walks if $c(t' + 1)$ is fixed. Finally, we use the definition of the auxiliary functions $h_j(i)$, which equal the probability that a random walk starting at time $t' + 1$ from u_j 's i -th neighbor reaches a node from D .

For the claim's second statement, we do a similar calculation for the process $\mathcal{SW}(t')$. By definition, this process consists already from time t' onwards of a collection of independent random walks. Thus, we can swap the expectation and the product in the last term of the above calculation, yielding the desired result. \square

Lemma 9. *Consider a time $t \geq 0$, a token j , and node v . Moreover, let $B \subseteq [\gamma n]$ and $D \subseteq V$ be arbitrary subset of tokens and nodes, respectively. Then, the following holds:*

- (a) $\Pr(w_j^S(t) = v) = \Pr(w_j^W(t) = v)$ and
(b) $\Pr\left(\bigcap_{j \in B} (w_j^S(t) \in D)\right) \leq \Pr\left(\bigcap_{j \in B} (w_j^W(t) \in D)\right)$
 $= \prod_{j \in B} \Pr(w_j^W(t) \in D).$

Proof. The first statement follows immediately from the definition of our process. For the second statement, note that the equality on the right-hand side holds trivially, since the tokens perform independent random walks in \mathcal{W} . To show the inequality, we define intermediate processes $\mathcal{SW}(t')$ ($t' \leq t$) that perform t' steps of \mathcal{S} followed by $t - t'$ steps of \mathcal{W} . By this definition, $\mathcal{SW}(0)$ is identical to \mathcal{W} restricted to t steps and, similar, $\mathcal{SW}(t)$ is identical to \mathcal{S} restricted to t steps. Consider the events

$$\mathcal{E}_{t'} := \bigcap_{j \in B} \left(w_j^{\mathcal{SW}(t')}(t) \in D \right), \quad (6)$$

stating that all tokens from B end up at nodes from D under process $\mathcal{SW}(t')$. The lemma's statement is equivalent to $\Pr(\mathcal{E}_t) \leq \Pr(\mathcal{E}_0)$. To prove this, we show $\Pr(\mathcal{E}_{t'+1}) \leq \Pr(\mathcal{E}_{t'})$ for all $t' \in \{0, 1, \dots, t-1\}$. Combining these inequalities yields the desired result.

Fix an arbitrary $t' \in \{0, 1, \dots, t-1\}$ and note that $\mathcal{SW}(t')$ and $\mathcal{SW}(t'+1)$ behave identical up to and including step t' . Hence, we can fix an arbitrary configuration (i.e., the location of each token) $c(t') = c$ immediately before time step $t' + 1$. For a token $j \in [\gamma n]$ let $u_j \in V$ denote its location in configuration c . Remember that $\pi_{u, t'+1}$ denote the (independent) random permutations chosen by each node u for time step $t' + 1$. To ease notation, we drop the time index $t' + 1$ and write π_u instead of $\pi_{u, t'+1}$ (and, similarly, $d(u)$ and $N(u, i)$ instead of $d_{t'+1}(u)$ and $N_{t'+1}(u, i)$). For each token j define a random variable $X_j \in [d(u_j) + 1]$ with $X_j = i$ if and only if $\pi_{u_j}(j) \in S_i$. In other words, X_j indicates to which of u_j 's neighbors token j is sent in time step $t' + 1$. We also introduce auxiliary functions $h_j: [d(u_j) + 1] \rightarrow [0, 1]$ defined by

$$h_j(i) := \Pr(w_j^W(t) \in D \mid w_j^W(t' + 1) = N(u_j, i)). \quad (7)$$

These are the probability that a random walk starting at time $t' + 1$ from u_j 's i -th neighbor ends up in a node from D . We can assume (w.l.o.g.) that all h_j are non-decreasing (by reordering the neighborhood of u_j accordingly). We show in Lemma 10 that the variables $(X_j)_{j \in B}$ satisfy the negative regression condition (cf. Definition 6). Additionally, Claim 8 relates the (conditioned) probabilities of the events $\mathcal{E}_{t'}$ and $\mathcal{E}_{t'+1}$ to the expectations over the different $h_j(X_j)$. With this, we get

$$\begin{aligned} \Pr(\mathcal{E}_{t'+1} \mid c(t') = c) &\stackrel{\text{Clm. 8(a)}}{=} \mathbb{E} \left[\prod_{j \in B} h_j(X_j) \mid c(t') = c \right] \\ &\stackrel{\text{Lem. 7}}{\leq} \prod_{j \in B} \mathbb{E}[h_j(X_j) \mid c(t') = c] \\ &\stackrel{\text{Clm. 8(b)}}{=} \Pr(\mathcal{E}_{t'} \mid c(t') = c). \end{aligned}$$

Using the law of total probability, we conclude $\Pr(\mathcal{E}_{t'+1}) \leq \Pr(\mathcal{E}_{t'})$, as required. \square

Lemma 10. *Fix a time $t' < t$ and an arbitrary configuration c . Let X_j be as in the proof of Lemma 9. Then the vector $(X_j)_{j \in [\gamma n]}$ satisfies the negative regression condition (NRC).*

Proof. Remember that u_j is the location of token j in configuration c and that $X_j \in [d_{t'+1}(u_j) + 1]$ indicates where token j is sent. We show for any $u \in V$ that $(X_j)_{j: u_j = u}$ satisfies the NRC. The lemma's statement follows since the π_u are chosen independently (if two independent vectors (X_j) and (Y_j) satisfy the NRC, then so do both together).

Fix a node u and disjoint subsets $\mathcal{L}, \mathcal{R} \subseteq \{j \in [\gamma n] \mid u_j = u\}$ of tokens on u . Define $d := d_{t'+1}(u)$ and let $f: [d+1]^{|\mathcal{L}|} \rightarrow \mathbb{R}$ be an arbitrary non-decreasing function. We have to show that $\mathbb{E}[f(X_l, l \in \mathcal{L}) \mid X_r = x_r, r \in \mathcal{R}]$ is non-increasing in each x_r (cf. Definition 6). That is, we need

$$\mathbb{E}[f(X_l, l \in \mathcal{L}) \mid X_r = x_r, r \in \mathcal{R}] \leq \mathbb{E}[f(X_l, l \in \mathcal{L}) \mid X_r = \tilde{x}_r, r \in \mathcal{R}], \quad (8)$$

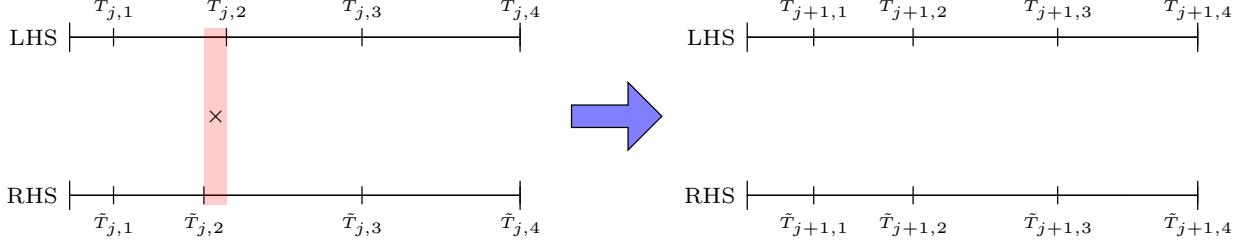


Figure 1: Illustration showing the $d+1 = 4$ different slots for the LHS and RHS process and how they change. In this example, $x_{\hat{r}} = 3$ and $\tilde{x}_{\hat{r}} = 2$. On the left, the uniform random variable U_j falls into slot $[T_1, T_2)$ for the LHS process (causing j to be sent to node $N(u, 2)$) and into slot $[\tilde{T}_2, \tilde{T}_3)$ for the RHS process (causing j to be sent to node $N(u, 3)$).

where $x_r = \tilde{x}_r$ holds for all $r \in \mathcal{R} \setminus \{\hat{r}\}$ and $x_{\hat{r}} > \tilde{x}_{\hat{r}}$ for a fixed index $\hat{r} \in \mathcal{R}$. We prove Inequality (8) via a coupling of the processes on the left-hand side (LHS process) and right-hand side (RHS process) of that inequality. Since $x_{\hat{r}} \neq \tilde{x}_{\hat{r}}$, these processes involve two slightly different probability spaces Ω and $\tilde{\Omega}$, respectively. To couple these, we employ a common uniform random variable $U_i \in [0, 1)$. By partitioning $[0, 1)$ into $d+1$ suitable slots for each process (corresponding to the slots S_i from the definition of \mathcal{S}), we can use the outcome of U_i to set the X_j in both Ω and $\tilde{\Omega}$. We first explain how to handle the case $x_{\hat{r}} - \tilde{x}_{\hat{r}} = 1$. The case $x_{\hat{r}} - \tilde{x}_{\hat{r}} > 1$ follows from this by a simple reordering argument.

So assume $x_{\hat{r}} - \tilde{x}_{\hat{r}} = 1$. We reveal the yet unset random variables X_j (i.e., $j \in [\gamma n] \setminus \mathcal{R}$) one by one in order of increasing indices. To ease the description assume (w.l.o.g.) that the tokens from \mathcal{R} are numbered from 1 to $|\mathcal{R}|$. When we reveal the j -th variable (which indicates the new location of the j -th token), note that the probability $p_{j,i}$ that token j is assigned to $N(u, i)$ depends solely on the *number* of previous tokens $j' < j$ that were assigned to $N(u, i)$. Thus, we can consider $p_{j,i}: \mathbb{N} \rightarrow [0, 1]$ as a function mapping $x \in \mathbb{N}$ to the probability that j is assigned to $N(u, i)$ conditioned on the event that exactly x previous tokens were assigned to $N(u, i)$. Note that $p_{j,i}$ is non-increasing. For a vector $\mathbf{x} \in \mathbb{N}^{d+1}$, we define a threshold function $T_{j,i}: \mathbb{N}^{d+1} \rightarrow [0, 1]$ by $T_{j,i}(\mathbf{x}) := \sum_{i' \leq i} p_{j,i'}(x_{i'})$ for each $i \in [d+1]$. To define our coupling, let $\beta_{j,i} := |\{j' < j \mid X_{j'} = i\}|$ denote the number of already revealed variables with value i in the LHS process and define, similarly, $\tilde{\beta}_{j,i} := |\{j' < j \mid \tilde{X}_{j'} = i\}|$ for the RHS process. We use $\boldsymbol{\beta}_j, \tilde{\boldsymbol{\beta}}_j \in \mathbb{N}^{d+1}$ to denote the corresponding vectors. Now, to assign token j we consider a uniform random variable $U_j \in [0, 1)$ and assign j in both processes using customized partitions of the unit interval. To this end, let $T_{j,i} := T_{j,i}(\boldsymbol{\beta}_j)$ and $\tilde{T}_{j,i} := T_{j,i}(\tilde{\boldsymbol{\beta}}_j)$ for each $i \in [d+1]$. We assign X_j in the LHS and RHS process as follows:

- **LHS Process:** $X_j = x_j = i$ if and only if $U_j \in [T_{j,i-1}, T_{j,i})$,
- **RHS Process:** $X_j = \tilde{x}_j = i$ if and only if $U_j \in [\tilde{T}_{j,i-1}, \tilde{T}_{j,i})$.

See Figure 1 for an illustration. Our construction guarantees that, considered in isolation, both the LHS and RHS process behave correctly.

At the beginning of this coupling, only the variables X_r corresponding to tokens $r \in \mathcal{R}$ are set, and these differ in the LHS and RHS process only for the index $\hat{r} \in \mathcal{R}$, for which we have $X_{\hat{r}} = x_{\hat{r}}$ (LHS) and $X_{\hat{r}} = \tilde{x}_{\hat{r}} = x_{\hat{r}} - 1$ (RHS). Let $\iota := x_{\hat{r}}$. For the first revealed token $j = \hat{r} + 1$, this implies $\beta_{j,\iota} = \tilde{\beta}_{j,\iota} + 1$, $\beta_{j,\iota-1} = \tilde{\beta}_{j,\iota-1} - 1$, and $\beta_{j,i} = \tilde{\beta}_{j,i}$ for all $i \notin \{\iota, \iota-1\}$. By the definitions of the slots for both processes, we get $T_{j,i} = \tilde{T}_{j,i}$ for all $i \neq \iota-1$ and $T_{j,\iota-1} > \tilde{T}_{j,\iota-1}$ (cf. Figure 1). Thus, the LHS and RHS process behave different if and only if $U_i \in [\tilde{T}_{j,\iota-1}, T_{j,\iota-1})$. If this happens, we get $x_j < \tilde{x}_j$ (i.e., token j is assigned to a smaller neighbor in the LHS process). This implies $\beta_{j+1} = \tilde{\beta}_{j+1}$ and both processes behave identical from now on. Otherwise, if $U_i \notin [\tilde{T}_{j,\iota-1}, T_{j,\iota-1})$, we have $\beta_{j+1} - \tilde{\beta}_{j+1} = \beta_j - \tilde{\beta}_j$ and we can repeat the above argument. Thus, after all X_j are revealed, there is at most one $j \in \mathcal{L}$ for which $x_j \neq \tilde{x}_j$, and for this we have $x_j < \tilde{x}_j$. Since f is non-decreasing, this guarantees Inequality (8). To handle the case $x_{\hat{r}} - \tilde{x}_{\hat{r}} > 1$, note that we can reorder the slots $[T_{j,i-1}, T_{j,i})$ used for the assignment of the variables such that the slots for $x_{\hat{r}}$ and $\tilde{x}_{\hat{r}}$ are neighboring. Formally, this merely changes in which order we consider the neighbors in the definition of the functions $T_{j,i}$. With this change, the same arguments as above apply. \square

With the above tools, we finally are able to prove the Chernoff bound stated in Lemma 4.

Proof of Lemma 4. Let $v_{j,t}$ denote the location of token j at time $(t-1) \cdot t_{\text{mix}}$. For all $t \leq T$ and $\ell \in \mathbb{N}$ define the random indicator variable $Y_{j,t}$ to be 1 if and only if the random walk starting at $v_{j,t}$ is at node u after t_{mix} time steps. By Lemma 9 we have for each $B' \subseteq B$ and $t \leq T$ that

$$\Pr\left(\bigcap_{i \in B'} X_{j,t} = 1\right) \leq \prod_{j \in B'} \Pr(Y_{j,t} = 1). \quad (9)$$

Hence for all $t \leq T$ and $\ell \in \mathbb{N}$ we have $\Pr\left(\sum_{j \in B} X_{j,t} \geq \ell\right) \leq \Pr\left(\sum_{j \in B} Y_{j,t} \geq \ell\right)$ and

$$\Pr(X \geq \ell) = \Pr\left(\sum_{t \leq T} \sum_{j \in B} X_{j,t} \geq \ell\right) \leq \Pr\left(\sum_{t \leq T} \sum_{j \in B} Y_{j,t} \geq \ell\right). \quad (10)$$

Let us define $p := 1/n + 1/n^5$. By the definition of t_{mix} , we have for all $j \in B$ and $t \leq T$ that

$$\Pr(Y_{j,t} = 1 \mid Y_{1,1}, Y_{2,1}, \dots, Y_{|B|,1}, Y_{1,2}, \dots, Y_{j-1,t}) \leq p. \quad (11)$$

Combining our observations with Lemma 11 (see below), we get $\Pr(X \geq \ell) \leq \text{Bin}(T \cdot |B|, p)$. Recall that $\mu = T \cdot |B| \cdot p$. Thus, by applying standard Chernoff bounds we get

$$\Pr(X \geq (1 + \delta)\mu) \leq \left(\frac{e^\delta}{(1 + \delta)^{1 + \delta}}\right)^\mu \leq e^{\delta^2 \mu / 3}, \quad (12)$$

which yields the desired statement. \square

Lemma 11 ([7, Lemma 3.1]). *Let X_1, X_2, \dots, X_n be a sequence of random variables with values in an arbitrary domain and let Y_1, Y_2, \dots, Y_n be a sequence of binary random variables with the property that $Y_i = Y_i(X_1, \dots, X_i)$. If $\Pr(Y_i = 1 \mid X_1, \dots, X_{i-1}) \leq p$, then*

$$\Pr\left(\sum Y_i \geq \ell\right) \leq \Pr(\text{Bin}(n, p) \geq \ell) \quad (13)$$

and, similarly, if $\Pr(Y_i = 1 \mid X_1, \dots, X_{i-1}) \geq p$, then

$$\Pr\left(\sum Y_i \leq \ell\right) \leq \Pr(\text{Bin}(n, p) \leq \ell). \quad (14)$$

Here, $\text{Bin}(n, p)$ denotes the binomial distribution with parameters n and p .

4 Protocol BALANCE

Protocol Description. The idea of our BALANCE protocol is quite simple: Every node u stores a k -dimensional vector $\ell_t(u)$ with k integer entries, one for each opinion. BALANCE simply performs an entry-wise load balancing on $\ell_t(u)$ according to the communication pattern $\mathbf{M} = (\mathbf{M}_t)_{t \leq N}$ and the corresponding transition matrices \mathbf{P}_t (cf. Section 2). Once the load is properly balanced, the nodes look at their largest entry and assume that this is the plurality opinion (stored in the variable plu_u).

In order to ensure a low memory footprint, we must not send fractional loads over the active edges. To this end, we use a rounding scheme from [11, 35], which works as follows: Consider a dimension $i \in [k]$ and let $\ell_{i,t}(u) \in \mathbb{N}$ denote the current (integral) load at u in dimension i . Then u sends $\lfloor \ell_{i,t}(u) \cdot \mathbf{P}_t[u, v] \rfloor$ tokens to all neighbors v with $\mathbf{M}_t[u, v] = 1$. This results in at most $d_t(u)$ remaining *excess tokens* ($\ell_{i,t}(u)$ minus the total number of tokens sent out). These are then randomly distributed (without replacement), where neighbor v receives a token with probability $\mathbf{P}_t[u, v]$. In the following we call the resulting balancing algorithm *vertex-based balancing* algorithm. The formal description of protocol BALANCE is given in Listing 2.

Analysis of BALANCE. Consider initial load vectors ℓ_0 with $\|\ell_0\|_\infty \leq n^5$. Let $\tau := \tau(g, \mathbf{M})$ be the first time step when VERTEX-BASED BALANCER under the (fixed) communication pattern $\mathbf{M} = (\mathbf{M}_t)_{t \leq N}$ is able to balance any such vector ℓ_0 up to a g -discrepancy (i.e., the minimal t with $\text{disc}(\ell_t) \leq g$). With these definitions, one can easily prove the following theorem.

```

1 for  $i \in [k]$ :
2   for  $\{u, v\} \in E$  with  $\mathbf{M}_t[u, v] = 1$ :
3     send  $\lfloor \ell_{i,t}(u) \cdot \mathbf{P}_t[u, v] \rfloor$  tokens from dimension  $i$  to  $v$ 
4      $x := \ell_{i,t}(u) - \sum_{v: \mathbf{M}_t[u, v] = 1} \lfloor \ell_{i,t}(u) \cdot \mathbf{P}_t[u, v] \rfloor$     {excess tokens}
5     randomly distribute  $x$  tokens such that:
6       every  $v \neq u$  with  $\mathbf{M}_t[u, v] = 1$  receives 1 token w.p.  $\mathbf{P}_t[u, v]$ 
7       (and zero otherwise)
8    $plu_u := i$  with  $\ell_{i,t}(u) \geq \ell_{j,t}(u) \quad \forall 1 \leq i, j \leq k$     {plurality guess}

```

Listing 2: Protocol BALANCE as executed by node u at time t . At time zero, each node initializes $\ell_{o_u,0}(u) := \gamma$ and $\ell_{j,0}(u) := 0$ for all $j \neq o_u$.

Theorem 12. Let $\alpha = \frac{n_1 - n_2}{n} \in [1/n, 1]$ denotes the initial bias. Consider a fixed communication pattern $\mathbf{M} = (\mathbf{M}_t)_{t \leq N}$ and an integer $\gamma \in [3 \cdot \frac{\gamma}{\alpha}, n^5]$. Protocol BALANCE ensures that all nodes know the plurality opinion after $\tau(g, \mathbf{M})$ rounds and requires $k \cdot \log(\gamma)$ memory bits per node.

Proof. Recall that $\gamma \geq 3 \frac{\gamma}{\alpha} = 3g \cdot \frac{n}{n_1 - n_2}$. For $i \in [k]$ let $\bar{\ell}_i := n_i \cdot \gamma / n$. The definition of $\tau(g, \mathbf{M})$ implies $\ell_{1,t}(u) \geq \bar{\ell}_1 - g$ and $\ell_{i,t}(u) \leq \bar{\ell}_i + g$ for all nodes u and $i \geq 2$. Consequently, we get

$$\ell_{1,t}(u) - \ell_{i,t}(u) \geq \bar{\ell}_1 - \bar{\ell}_i - 2g = 3g \cdot \frac{n_1 - n_i}{n_1 - n_2} - 2g > 0. \quad (15)$$

Thus, every node u has the correct plurality guess at time t . \square

The memory usage of BALANCE depends on the number of opinions (k) and on the number of tokens generated on every node (γ). The algorithm is very efficient for small values of k but it becomes rather impractical if k is large. Note that if one chooses γ sufficiently large, it is easy to adjust the algorithm such that every node knows the frequency of *all* opinions in the network. The following corollary gives a few concrete examples for common communication patterns on general graphs.

Corollary 13. Let G be an arbitrary d -regular graph. BALANCE ensures that all nodes agree on the plurality opinion with probability $1 - e^{-(\log(n))^c}$ for some constant c

- (a) using $O(k \cdot \log n)$ bits of memory in time $O(\frac{\log n}{1 - \lambda_2})$ in the diffusion model,
- (b) using $O(k \cdot \log n)$ bits of memory in time $O(\frac{1}{d \cdot p_{\min}} \cdot \frac{\log n}{1 - \lambda_2})$ in the random matching model,
- (c) using $O(k \cdot \log(\alpha^{-1}))$ bits of memory in time $O(d \cdot \frac{\log n}{1 - \lambda_2})$ in the balancing circuit model, and
- (d) using $O(k \cdot \log(\alpha^{-1}))$ bits of memory in time $O(n \cdot \frac{\log n}{1 - \lambda_2})$ in the sequential model.

Proof. Part (a) follows directly from [36, Theorem 6.6] and Part (c) follows directly from [36, Theorem 1.1]. To show Part (b) and (d) we choose τ such that $\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_\tau$ enable VERTEX-BASED BALANCER to balance any vector ℓ_0 (with initial discrepancy of at most n^5) up to a g -discrepancy. The bound on τ then follows from [36, Theorem 1.1]. \square

In particular, for the complete graph and $k = 2$, Corollary 13(d) gives the same bounds as the bound from [3]. Note that the s states used to measure space requirement in [3] correspond to $\log s$ memory bits in our model.

References

- [1] Mohammed Amin Abdullah and Moez Draief. Global majority consensus by local majority polling on graphs of a given degree sequence. *Discrete Applied Mathematics*, 180:1–10, 2015.
- [2] D. Aldous and J. Fill. Reversible markov chains and random walks on graphs, 2002. Unpublished. <http://www.stat.berkeley.edu/~aldous/RWG/book.html>.

- [3] Dan Alistarh, Rati Gelashvili, and Milan Vojnovic. Fast and exact majority in population protocols. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, (PODC)*, pages 47–56, 2015.
- [4] Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, 2008.
- [5] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.
- [6] James Aspnes and Eric Ruppert. An introduction to population protocols. *Bulletin of the EATCS*, 93:98–117, 2007.
- [7] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM Journal of Computing*, 29(1):180–200, 1999.
- [8] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Riccardo Silvestri. Plurality consensus in the gossip model. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 371–390, 2015.
- [9] Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, Riccardo Silvestri, and Luca Trevisan. Simple dynamics for plurality consensus. In *26th ACM Symposium on Parallelism in Algorithms and Architectures, (SPAA)*, pages 247–256, 2014.
- [10] Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Stabilizing consensus with many opinions. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2016.
- [11] Petra Berenbrink, Colin Cooper, Tom Friedetzky, Tobias Friedrich, and Thomas Sauerwald. Randomized diffusion for indivisible loads. *J. Comput. Syst. Sci.*, 81(1):159–185, 2015.
- [12] Stephen P. Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.
- [13] Luca Cardelli and Attila Csikász-Nagy. The cell cycle switch computes approximate majority. *Scientific reports*, 2, 2012.
- [14] Yuan-Jyue Chen, Neil Dalchau, Niranjana Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from dna. *Nature nanotechnology*, 8(10):755–762, 2013.
- [15] Andrea E. F. Clementi, Miriam Di Ianni, Giorgio Gambosi, Emanuele Natale, and Riccardo Silvestri. Distributed community detection in dynamic graphs. *Theor. Comput. Sci.*, 584:19–41, 2015.
- [16] Colin Cooper, Robert Elsässer, Hirotaka Ono, and Tomasz Radzik. Coalescing random walks and voting on connected graphs. *SIAM J. Discrete Math.*, 27(4):1748–1758, 2013.
- [17] Colin Cooper, Robert Elsässer, and Tomasz Radzik. The power of two choices in distributed voting. In *Automata, Languages, and Programming - 41st International Colloquium, (ICALP)*, pages 435–446, 2014.
- [18] Benjamin Doerr, Leslie Ann Goldberg, Lorenz Minder, Thomas Sauerwald, and Christian Scheideler. Stabilizing consensus with the power of two choices. In *Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, (SPAA)*, pages 149–158, 2011.
- [19] Peter Donnelly and Dominic Welsh. Finite particle systems and infection models. *Mathematical Proceedings of the Cambridge Philosophical Society*, 94(1):167–182, 1983.
- [20] Moez Draief and Milan Vojnovic. Convergence speed of binary interval consensus. *SIAM J. Control and Optimization*, 50(3):1087–1109, 2012.

- [21] Devdatt P. Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *Random Struct. Algorithms*, 13(2):99–124, 1998.
- [22] Bhaskar Ghosh and S. Muthukrishnan. Dynamic load balancing by random matchings. *J. Comput. Syst. Sci.*, 53(3):357–370, 1996.
- [23] Yehuda Hassin and David Peleg. Distributed probabilistic polling and applications to proportionate agreement. *Inf. Comput.*, 171(2):248–268, 2001.
- [24] R. Holley and T. Liggett. Ergodic theorems for weakly interacting infinite systems and the voter model. *The Annals of Probability*, 3(4):643–663, 1975.
- [25] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 482–491, 2003.
- [26] N. Lanchier and C. Neuhauser. Voter model and biased voter model in heterogeneous environments. *Journal of Applied Probability*, 44(3):770–787, 2007.
- [27] Thomas Liggett. *Interacting particle systems*. Springer Science & Business Media, 2012.
- [28] F. Mallmann-Trenn. Bounds on the voting time in terms of the conductance. Master’s thesis, Simon Fraser University, 2014. Master’s thesis. <http://summit.sfu.ca/item/14502>.
- [29] George B. Mertzios, Sotiris E. Nikolettseas, Christoforos Raptopoulos, and Paul G. Spirakis. Determining majority in networks with local interactions and very small local memory. In *Automata, Languages, and Programming - 41st International Colloquium, (ICALP)*, pages 871–882, 2014.
- [30] Elchanan Mossel, Joe Neeman, and Omer Tamuz. Majority dynamics and aggregation of information in social networks. *Autonomous Agents and Multi-Agent Systems*, 28(3):408–429, 2014.
- [31] Elchanan Mossel and Grant Schoenebeck. Reaching consensus on social networks. In *Innovations in Computer Science - (ICS)*, pages 214–229, 2010.
- [32] David Peleg. Local majorities, coalitions and monopolies in graphs: a review. *Theor. Comput. Sci.*, 282(2):231–257, 2002.
- [33] Etienne Perron, Dinkar Vasudevan, and Milan Vojnovic. Using three states for binary consensus on complete graphs. In *28th IEEE International Conference on Computer Communications, (INFOCOM)*, pages 2527–2535, 2009.
- [34] Yuval Rabani, Alistair Sinclair, and Rolf Wanka. Local divergence of markov chains and the analysis of iterative load balancing schemes. In *39th Annual Symposium on Foundations of Computer Science, FOCS*, pages 694–705, 1998.
- [35] Thomas Sauerwald and He Sun. Tight bounds for randomized load balancing on arbitrary network topologies. In *53rd Annual IEEE Symposium on Foundations of Computer Science, (FOCS)*, pages 341–350, 2012.
- [36] Thomas Sauerwald and He Sun. Tight bounds for randomized load balancing on arbitrary network topologies, 2012. full version of FOCS’12.